End-to-End (E2E) Testing

System testing

Acceptance testing

Acceptence Testing

- **Testing the entire system as a whole.**
  - **UI + Server-side + Database**

- **Concerns:**
  - **Functionality ****
    - **From the USER interface perspective.**
  - **Performance**
  - **Load/Stress**

E2E Testing

- **Many similarities to API testing:**
  - **Blackbox – not looking at internals, only expected output for specific inputs.**
    - **May also be interested in side-effects, e.g. database changes.**
  - **The Asynchronuous nature (for web/mobile apps).**

- **Unit and Integration test should have ironed out (most) 'low level' errors.**

E2E Testing

- **Web apps - Targeting the browser interface.**
  - **Form submits.**
  - **Navigation.**
  - **Flows e.g. shopping cart checkout.**

Automation Tools

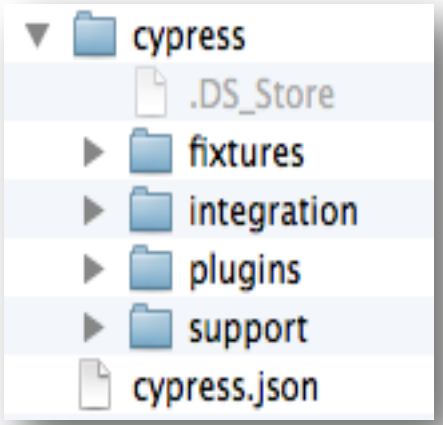
- **Traditional tool suite: Mocha + Chai +** Selenium**.**
- **(Very) modern tool suite:** Cypress
  - **Uses Mocha and Chai internally.**
- **Cypress.**
  - **Win / Mac / Linux**
  - **MIT License**
  - **Open Source**

Cypress

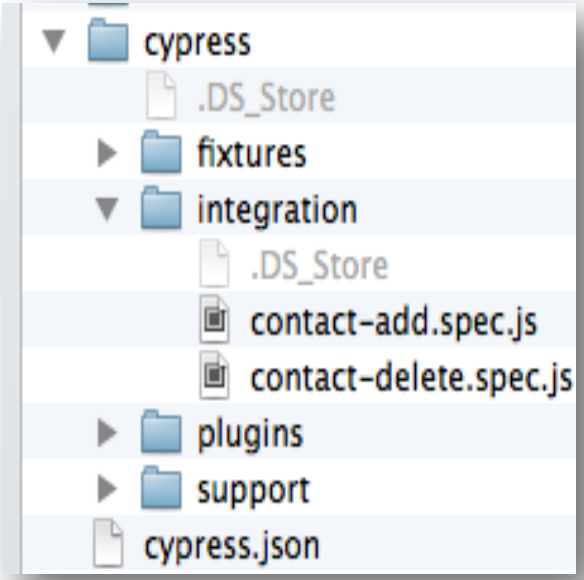
- **Getting started:**

  $ npm install –save-dev cypress

- **(Default) Test code folder structure:**



- **fixtures** – mock data
- **Integration** – test code, termed specs.
- **cypress.json** – config file

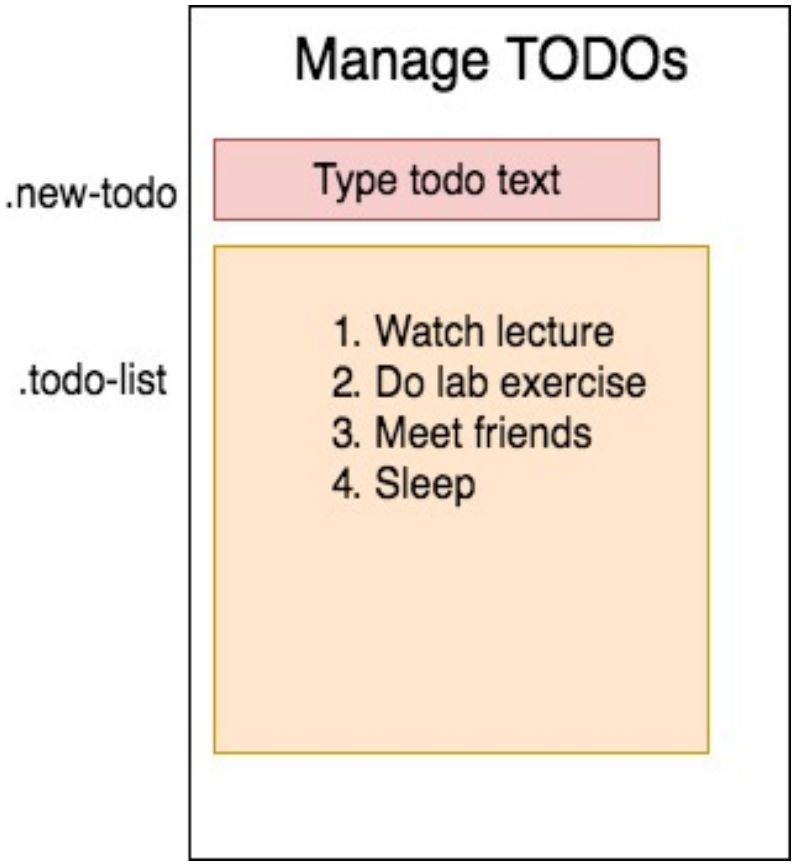


- **CLI (Command Line Interface) has 2 main commands:**

  $ npx cypress open **- GUI interactive mode (Dev)**

  $ npx cypress run **- headless mode (Testing/CI).**

Sample test code




```
describe("TODO app", () => {
    it('adds 2 todos', () => {
      cy.visit('http://localhost:3000')
      cy.get('.new-todo')
        .type('learn testing{enter}')
        .type('complete lab{enter}')
      cy.get('.todo-list li')
        .should('have.length', 2)
    })
})
```

- **Method Chaining style e.g.**
  cy(…)get(…).type(…)

Cypress statements

cy.get(. . .selector …).should(…assertion/expectation)

Command **Assertion** (Expectations)

- **Commands:**
  - **get() - Get one or more DOM elements by selector**
  - **contains(text) - Get the DOM element containing the text, e.g**. cy.contains('Welcome')
  - **find() - Get the descendent DOM elements of a selector.**

    e.g. cy.get('.article').find(button') // Yield the 'button' within '.article'
  - **click() – click a DOM element,** e.g. cy.get('button').click()
  - **select() - Select an <option> within a <select>**

    e.g. cy.get('#paymenttype').select('Visa')

  - See https://docs.cypress.io/api/api/table-of-contents.html

Cypress statements

cy.get(. . .selector …).should(…assertion/expectation)

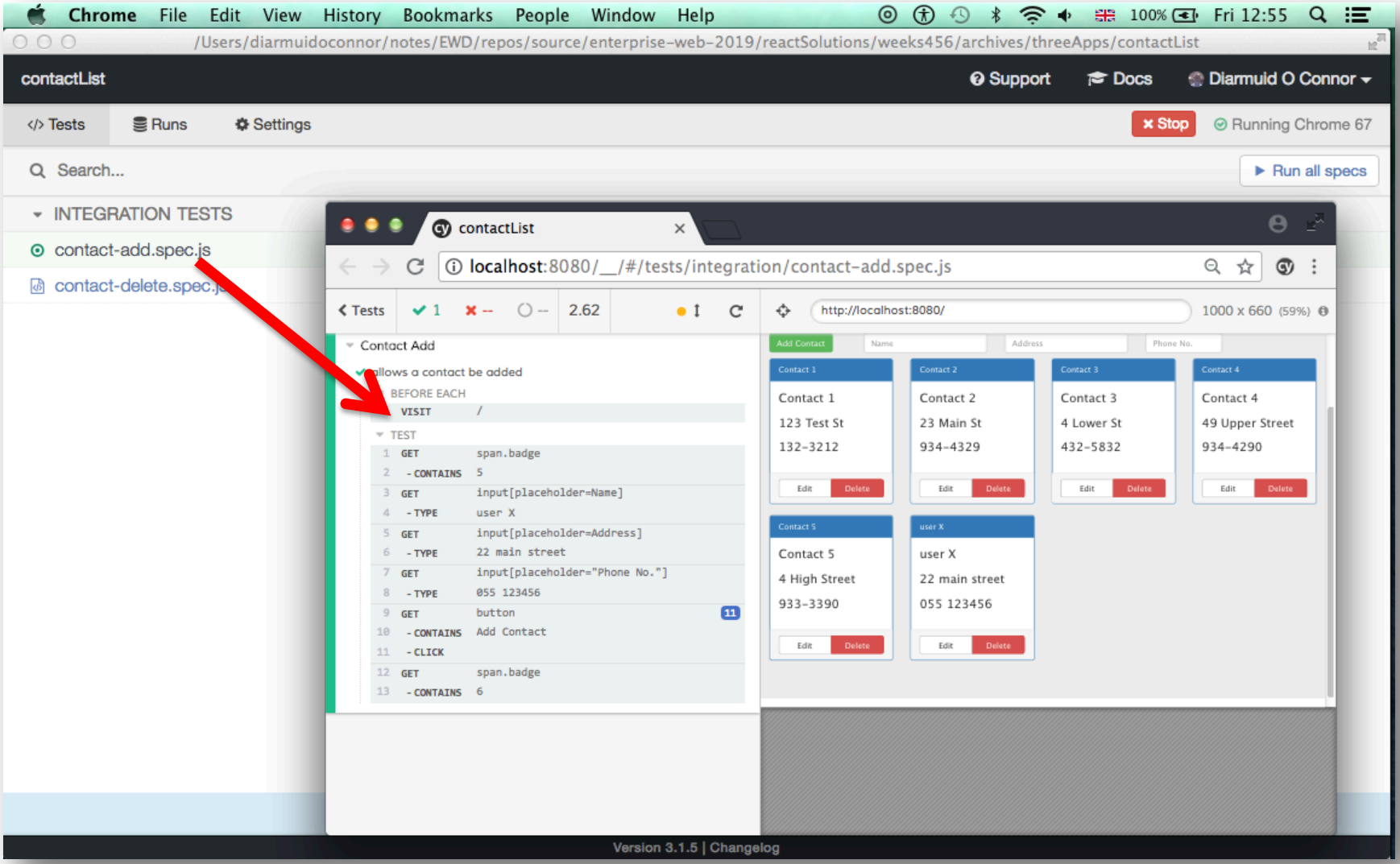
- **Selector: Based on CSS/JQuery style.**
  - –**Id,** e.g. cy.get(’#heading’)
  - –**CSS Class,** e.g. cy.get(’.info-message’)
  - –**Tag,** e.g. cy.get(’input’)
  - –**Attributes,** cy.get('button[type=submit]').click()
    - •**The** data-test **attribute.**
  - –**nth-child, e.g. get the 8th column of the 3rd row in a table**

    cy.get('tbody').find('tr:nth-child(3)').find('td:nth-child(8)’)
  - –**These can be combined, e.g.** div.container (the div tag with CSS class .container)

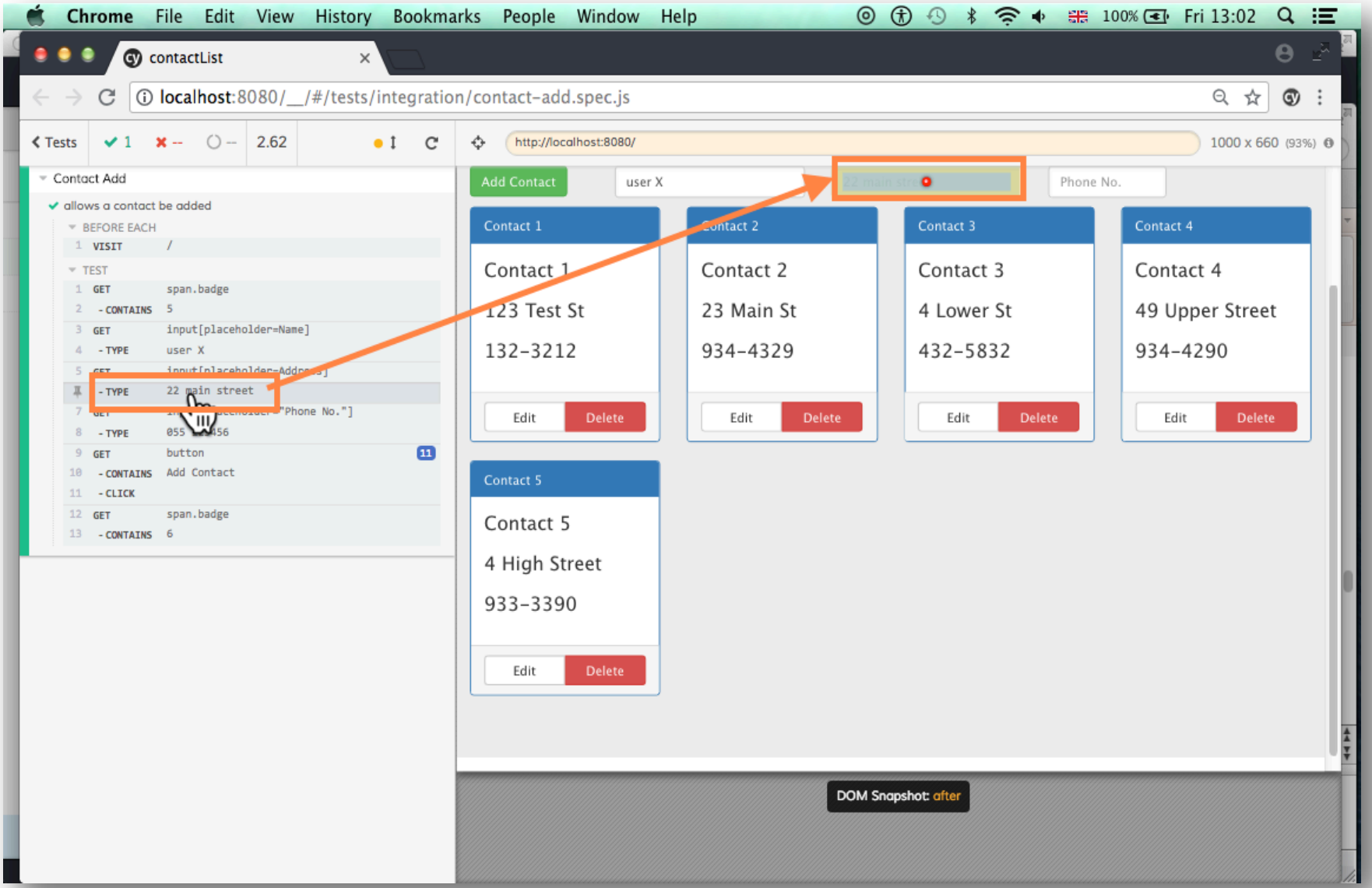
Cypress Test Runner

- **Main features:**
  - **Tests run** inside the browser.
    - **Full access to browser resources – DOM, cookies, local storage.**
    - **Framework-agnostic.**
  - Flake-free **test execution.**
    - **Auto retries commands (e.g. get()) to cope with slow DOM construction.**
    - **Deals with unpredictable nature of the web.**
  - **Supports** time-travel **for convenient debugging.**

$ npx cypress open (Interactive runner)



- **Time-travel – Step through test code to track UI state.**



Selectors

cy.get(. . .selector …).should(…assertion/expectation)

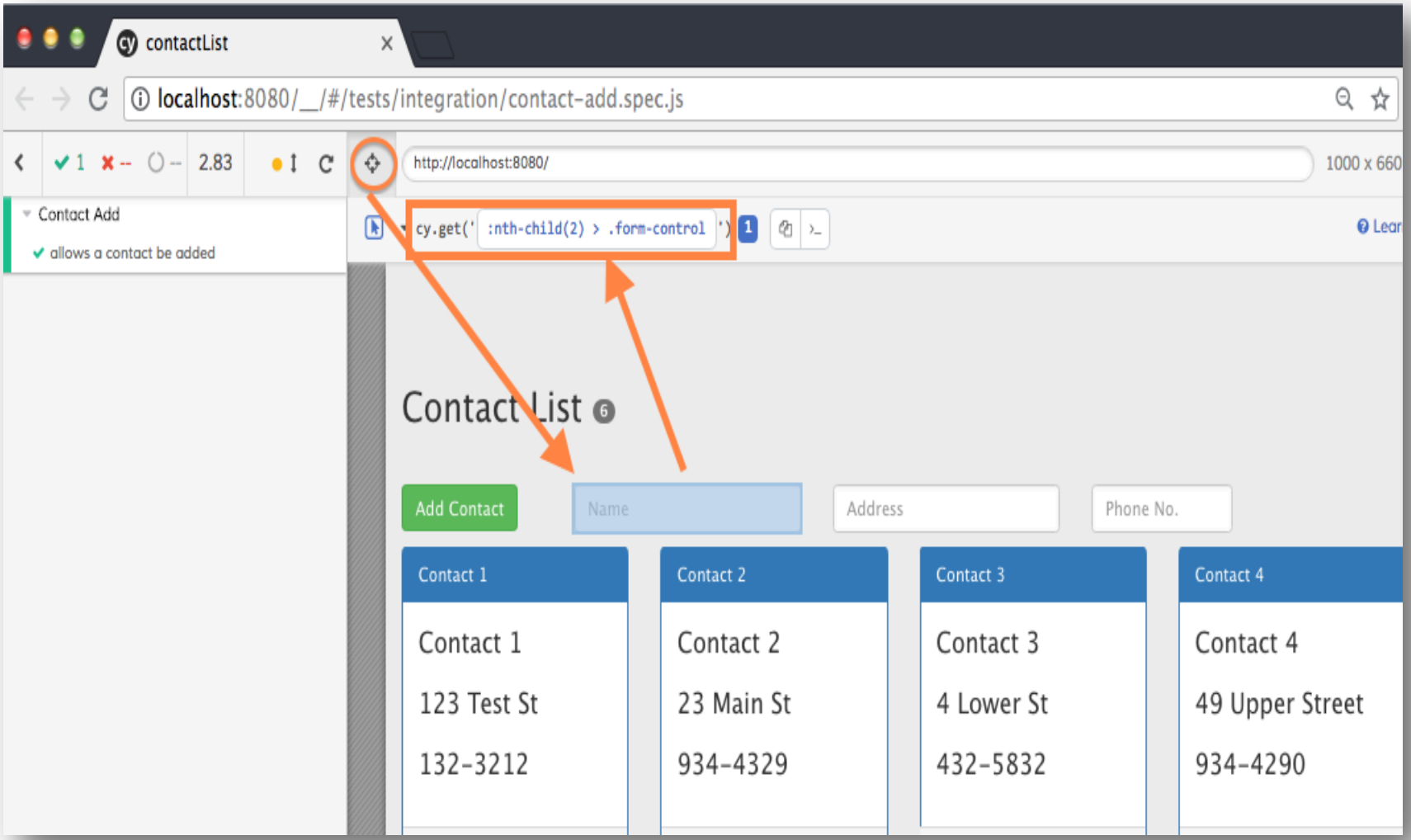
- **Has impact on test** brittleness.
  - **Small changes to CSS or HTNL structure can cause test failure.**

- **Use** data-test **attribute, where possible, to avoid brittle tests, e.g.**

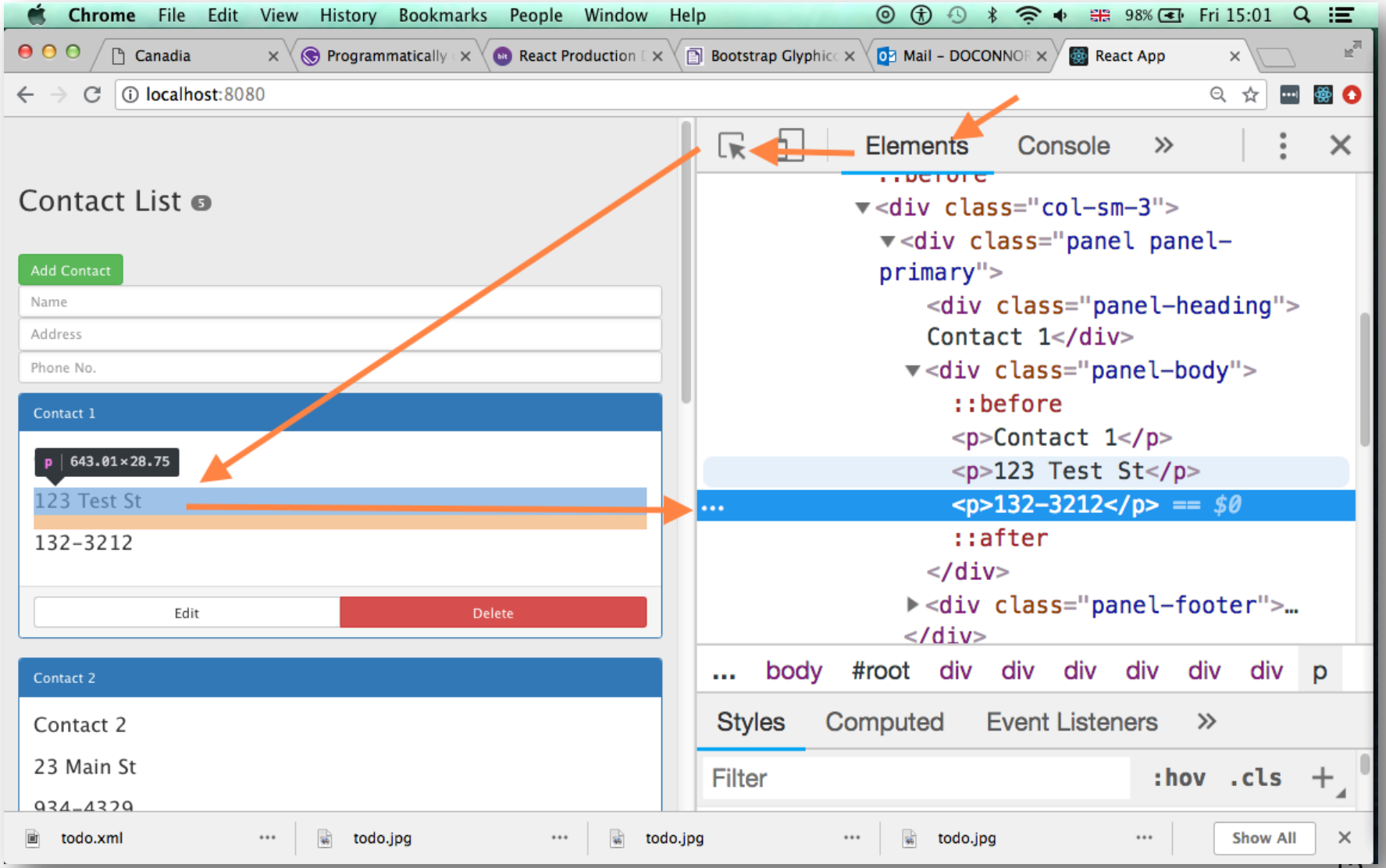
```
<input data-test=”name" type="text" className="form-control” onChange={….} />
```

```
cy.get('input[data-test=name]').type(“Joe Bloggs”)
```

- Selector Playground **– good learning aid.**



- **Use** Chrome dev tools **to also assist with choosing a selector.**



Cypress Test Runner

- **Headless mode:**

  $ npx cypress run

- **Runs all tests.**
- **Ideal for CI (Continuous Integration) environment.**
- **Generates video recordings (default; configurable).**
  - **.mp4 file type.**
  - **Facilitates sharing and project visibility.**
  - **Dashboard service (Publish recordings)**

Commands are in a queue

```
it('adds 2 todos', () => {
   cy.visit('http://localhost:3000')
   cy.get('.new-todo')
      .type('learn testing{enter}')
      .type('complete lab{enter}')
   cy.get('.todo-list li')
      .should('have.length', 2)
})
```